Министерство образования и науки РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

‘Распределенные системы хранения данных’

Вариант №38491

Выполнил:
Студент группы
Р33312
Соболев Иван
Александрович
Преподаватель:
Осипов Святослав
Владимирович

Санкт-Петербург, 2024

Задание:

Этап 1. Конфигурация

Настроить репликацию postgres на трех узлах в каскадном режиме A --> B --> C. Для управления использовать pgpool-II. Репликация с A на B синхронная. Репликация с B на C асинхронная. Продемонстрировать, что новые данные реплицируются на B в синхронном режиме, а на C с задержкой.

Этап 2. Симуляция и обработка сбоя

2.1 Подготовка:

- Установить несколько клиентских подключений к СУБД.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

2.2 Сбой:

1. Симулировать отказ основного узла - выполнить жесткое выключение виртуальной машины.

2.3 Обработка:

- Найти и продемонстрировать в логах релевантные сообщения об ошибках.
- Выполнить переключение (failover) на резервный сервер.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Восстановление

- Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.
- Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.
- Восстановить исправную работу узлов в исходной конфигурации (в соответствии с этапом 1).
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Выполнение:

Для выполнения сейчас и далее использовался Docker.

Настройка рабочего окружения

Создаем Docker образ, на который будем накатывать узлы:

```
#Base Image
FROM ubuntu:23.04


#Update APT repository & Install OpenSSH
RUN apt-get update \
   && apt-get install -y openssh-server \
   && apt-get install -y mysql-client \
   && apt-get -y install curl wget sudo \
   && apt-get -y install ca-certificates gnupg


#Postgres 15
ARG DEBIAN_FRONTEND=noninteractive


RUN sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
RUN wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
RUN apt-get -y install postgresql-15
RUN apt-get -y install pgpool2 libpgpool2 postgresql-15-pgpool2
RUN apt-get -y install ssh iputils-ping vim nano


RUN cp -s /usr/lib/postgresql/15/bin/* /usr/bin 2> dev/null; exit 0

#Postgres 15


##Establish the operating directory of OpenSSH

#RUN mkdir /var/run/sshd


#Set Root password

RUN echo 'root:root' | chpasswd


#Allow Root login

RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' \

   /etc/ssh/sshd_config


#SSH login fix

RUN sed 's@session\s*required\s*pam_loginuid.so@session optional \

   pam_loginuid.so@g' -i /etc/pam.d/sshd


#expose port 22

EXPOSE 22
```

```
#Commands to be executed by default
CMD ["/usr/sbin/sshd","-D"]
```

Описание узлов:

```
version: "3.6"


services:
 ubuntu-a:
   build:
     context: .
     dockerfile: Dockerfile
   hostname: ubuntu-a
   container_name: ubuntu-a
   networks:
     ubuntu-net:
 ubuntu-b:
   build:
     context: .
     dockerfile: Dockerfile
   hostname: ubuntu-b
   container_name: ubuntu-b
   networks:
```

```yaml
      ubuntu-net:
  ubuntu-c:
    build:
      context: .
      dockerfile: Dockerfile
    hostname: ubuntu-c
    container_name: ubuntu-c
    networks:
      ubuntu-net:
  ubuntu-pool:
    build:
      context: .
      dockerfile: Dockerfile
    hostname: ubuntu-pool
    container_name: ubuntu-pool
    networks:
      ubuntu-net:


networks:
  ubuntu-net:
    driver: bridge
```

Этап 1. Настройка

Создаем докер образ:

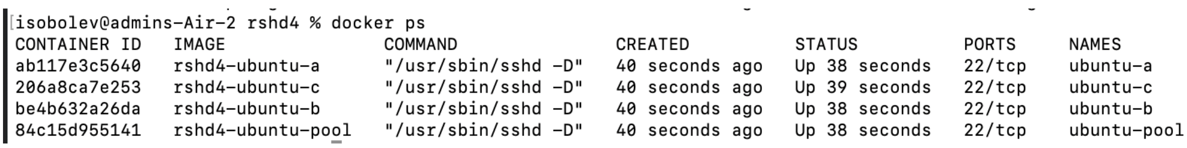
docker build -t rshd-postgres .

Создаем новую `bridge` сеть:

docker network create rshd-bridge

Запускаем сервисы:

docker compose up -d

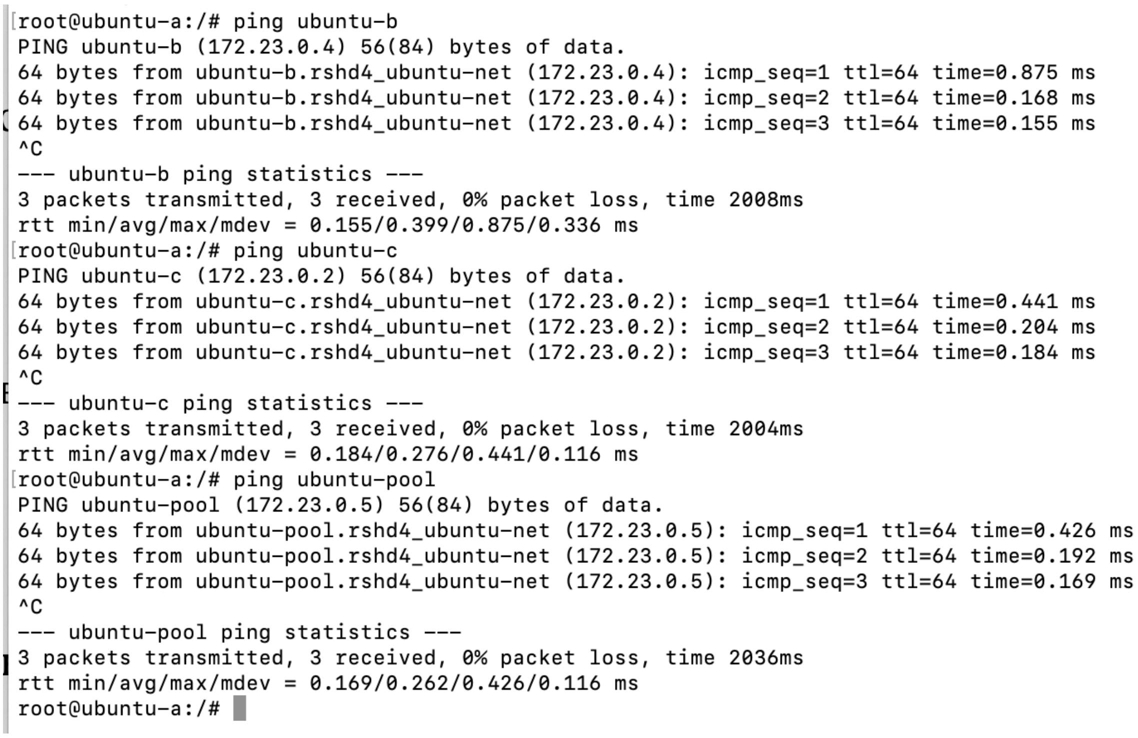


```
isobolev@admins-Air-2 rshd4 % docker ps
CONTAINER ID   IMAGE                COMMAND               CREATED          STATUS          PORTS     NAMES
ab117e3c5640   rshd4-ubuntu-a       "/usr/sbin/sshd -D"   40 seconds ago   Up 38 seconds   22/tcp    ubuntu-a
206a8ca7e253   rshd4-ubuntu-c       "/usr/sbin/sshd -D"   40 seconds ago   Up 39 seconds   22/tcp    ubuntu-c
be4b632a26da   rshd4-ubuntu-b       "/usr/sbin/sshd -D"   40 seconds ago   Up 38 seconds   22/tcp    ubuntu-b
84c15d955141   rshd4-ubuntu-pool    "/usr/sbin/sshd -D"   40 seconds ago   Up 38 seconds   22/tcp    ubuntu-pool
```

Подключение к запущенному контейнеру:

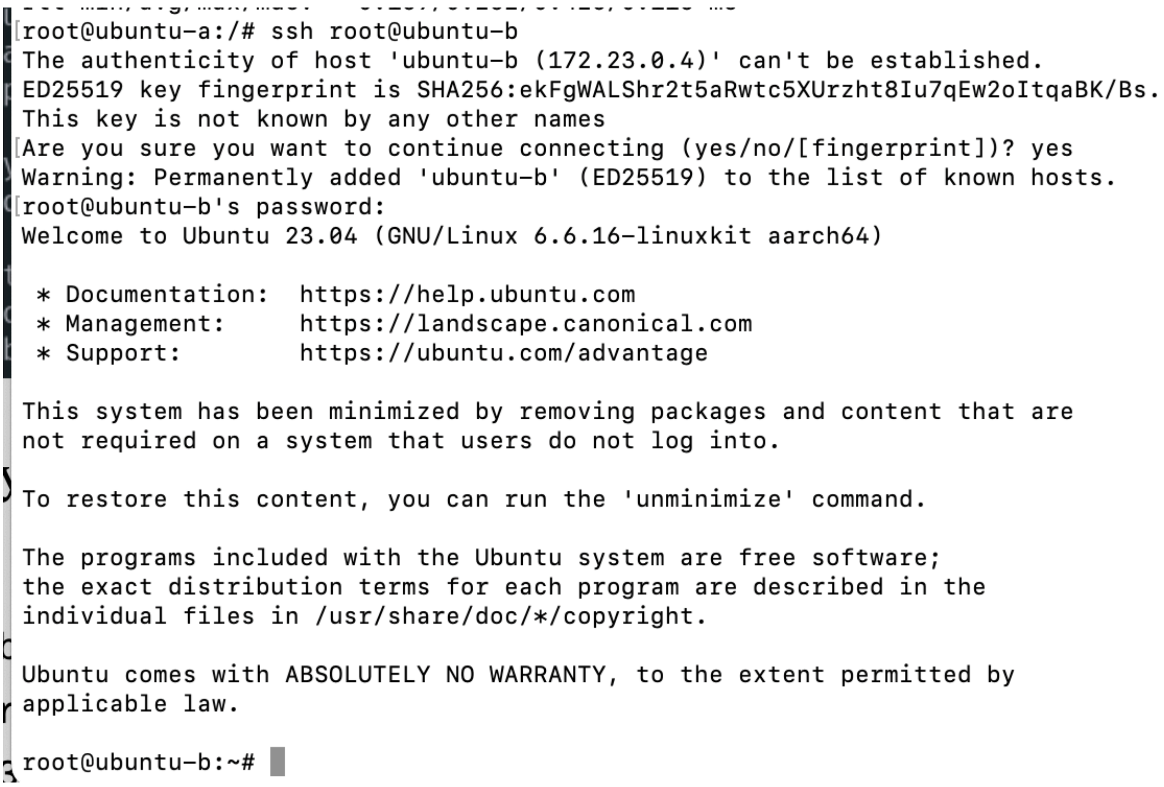
docker exec -it ubuntu-a bash

Проверка ping:



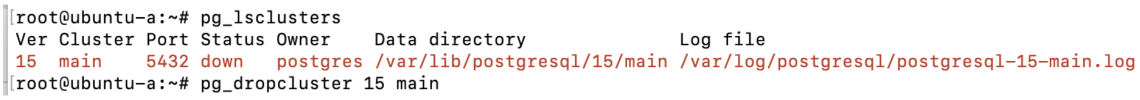
```
root@ubuntu-a:/# ping ubuntu-b
PING ubuntu-b (172.23.0.4) 56(84) bytes of data.
64 bytes from ubuntu-b.rshd4_ubuntu-net (172.23.0.4): icmp_seq=1 ttl=64 time=0.875 ms
64 bytes from ubuntu-b.rshd4_ubuntu-net (172.23.0.4): icmp_seq=2 ttl=64 time=0.168 ms
64 bytes from ubuntu-b.rshd4_ubuntu-net (172.23.0.4): icmp_seq=3 ttl=64 time=0.155 ms
^C
--- ubuntu-b ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2008ms
rtt min/avg/max/mdev = 0.155/0.399/0.875/0.336 ms
root@ubuntu-a:/# ping ubuntu-c
PING ubuntu-c (172.23.0.2) 56(84) bytes of data.
64 bytes from ubuntu-c.rshd4_ubuntu-net (172.23.0.2): icmp_seq=1 ttl=64 time=0.441 ms
64 bytes from ubuntu-c.rshd4_ubuntu-net (172.23.0.2): icmp_seq=2 ttl=64 time=0.204 ms
64 bytes from ubuntu-c.rshd4_ubuntu-net (172.23.0.2): icmp_seq=3 ttl=64 time=0.184 ms
^C
--- ubuntu-c ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.184/0.276/0.441/0.116 ms
root@ubuntu-a:/# ping ubuntu-pool
PING ubuntu-pool (172.23.0.5) 56(84) bytes of data.
64 bytes from ubuntu-pool.rshd4_ubuntu-net (172.23.0.5): icmp_seq=1 ttl=64 time=0.426 ms
64 bytes from ubuntu-pool.rshd4_ubuntu-net (172.23.0.5): icmp_seq=2 ttl=64 time=0.192 ms
64 bytes from ubuntu-pool.rshd4_ubuntu-net (172.23.0.5): icmp_seq=3 ttl=64 time=0.169 ms
^C
--- ubuntu-pool ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2036ms
rtt min/avg/max/mdev = 0.169/0.262/0.426/0.116 ms
root@ubuntu-a:/# 
```

Проверка ssh:



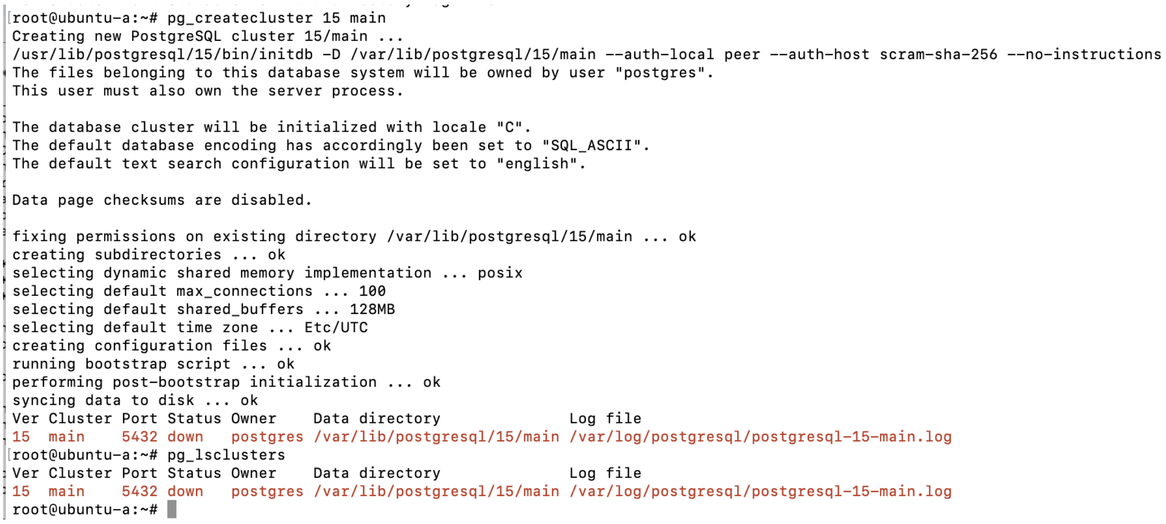
```
root@ubuntu-a:/# ssh root@ubuntu-b
The authenticity of host 'ubuntu-b (172.23.0.4)' can't be established.
ED25519 key fingerprint is SHA256:ekFgWALShr2t5aRwtc5XUrzht8Iu7qEw2oItqaBK/Bs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ubuntu-b' (ED25519) to the list of known hosts.
root@ubuntu-b's password:
Welcome to Ubuntu 23.04 (GNU/Linux 6.6.16-linuxkit aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@ubuntu-b:~#
```

Дропаем все существующие кластеры:



```
root@ubuntu-a:~# pg_lsclusters
Ver Cluster Port Status Owner    Data directory              Log file
15  main    5432 down   postgres /var/lib/postgresql/15/main /var/log/postgresql/postgresql-15-main.log
root@ubuntu-a:~# pg_dropcluster 15 main
```

Создаем новый кластер:



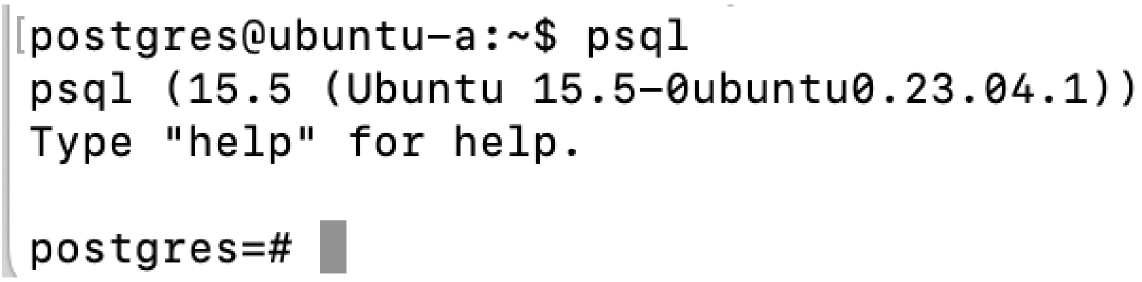
```
root@ubuntu-a:~# pg_createcluster 15 main
Creating new PostgreSQL cluster 15/main ...
/usr/lib/postgresql/15/bin/initdb -D /var/lib/postgresql/15/main --auth-local peer --auth-host scram-sha-256 --no-instructions
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "C".
The default database encoding has accordingly been set to "SQL_ASCII".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/15/main ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
Ver Cluster Port Status Owner    Data directory              Log file
15  main    5432 down   postgres /var/lib/postgresql/15/main /var/log/postgresql/postgresql-15-main.log
root@ubuntu-a:~# pg_lsclusters
Ver Cluster Port Status Owner    Data directory              Log file
15  main    5432 down   postgres /var/lib/postgresql/15/main /var/log/postgresql/postgresql-15-main.log
root@ubuntu-a:~#
```

Запускаем кластер:

pg_ctlcluster 15 main start

Подключаемся к пользователю `postgres`:

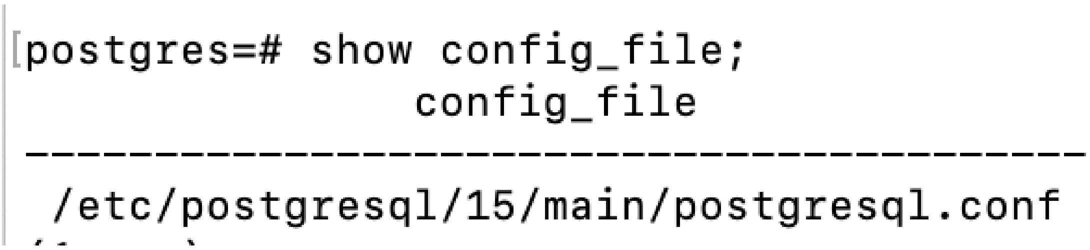
su - postgres



```
postgres@ubuntu-a:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# 
```

Конфигурация `postgresql.conf`:

Расположение файла конфигурации:



```
postgres=# show config_file;
               config_file
-----------------------------------------
 /etc/postgresql/15/main/postgresql.conf
```

UBUNTU-A

cluster_name = 'cluster_a'

listen_addresses = '*'

wal_level = replica

max_wal_senders = 10

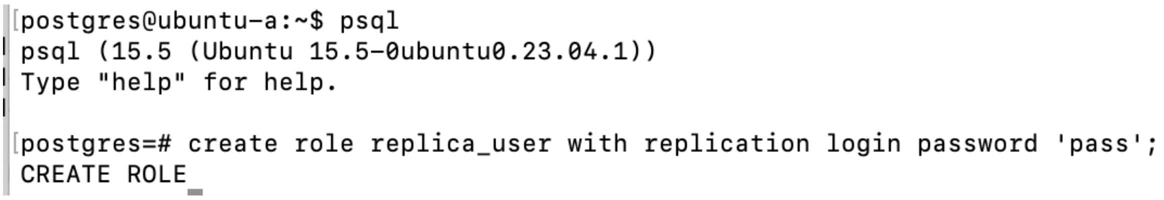
synchronous_standby_names = 'cluster_b'

synchronous_commit = on

hot_standby = on

Включаем параметр hot_standby. Данный параметр будет игнорироваться на master сервере. Но если master сервер станет slave сервером, то данный параметр будет необходим.

Создаем пользователя репликации:



```
postgres@ubuntu-a:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# create role replica_user with replication login password 'pass';
CREATE ROLE
```

Получаем айпишники нод:

PING ubuntu-a (172.23.0.3) 56(84) bytes of data.
PING ubuntu-b (172.23.0.4) 56(84) bytes of data.
PING ubuntu-c (172.23.0.2) 56(84) bytes of data.

Добавляем записи репликаций в `pg_hba.conf`:

```
host   replication   replica_user   172.23.0.4/24      md5
host   replication   replica_user   172.23.0.2/24      md5
```

UBUNTU-B

cluster_name = 'cluster_b'

listen_addresses = '*'

wal_level = replica

max_wal_senders = 10

hot_standby = on

Добавляем записи репликаций в `pg_hba.conf`:

```
host   replication   replica_user   172.23.0.3/24      md5
host   replication   replica_user   172.23.0.2/24      md5
```

UBUNTU-C

cluster_name = 'cluster_c'

listen_addresses = '*'

wal_level = replica

max_wal_senders = 10

hot_standby = on

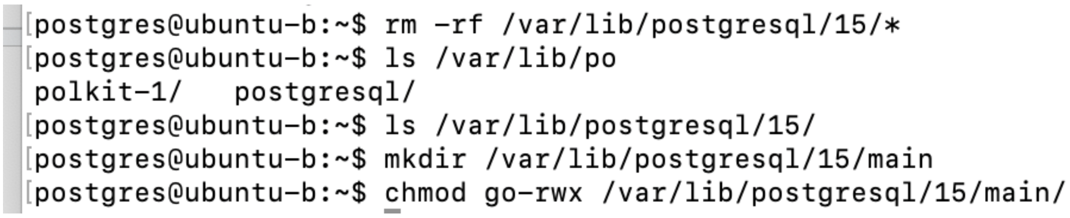
Добавляем записи репликаций в `pg_hba.conf`:

```
host    replication     replica_user    172.23.0.3/24          md5
host    replication     replica_user    172.23.0.4/24          md5
```

pg_basebackup

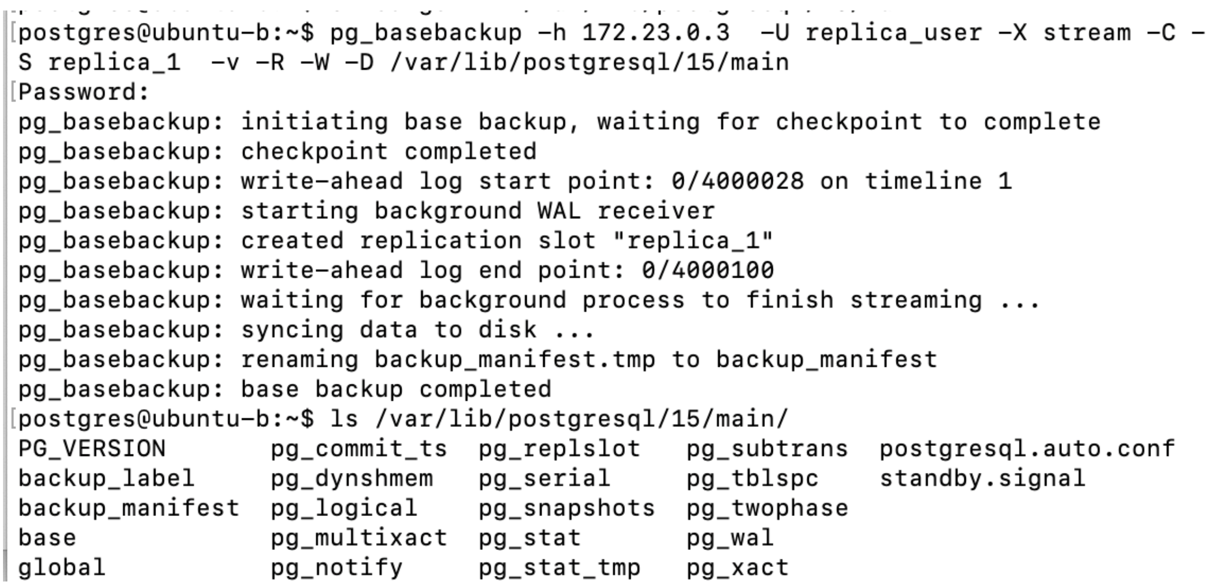
Осуществляем backup основного узла для того, что получить зеркало на текущий момент. Для каждого из secondary узлов удаляем `PGDATA` содержимое.

UBUNTU-B



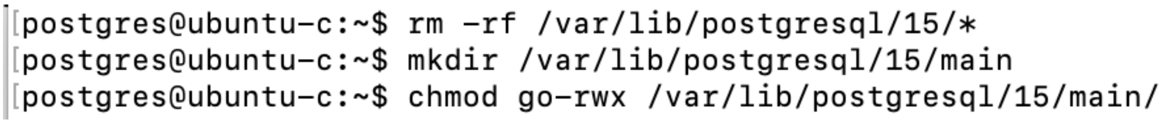
```
[postgres@ubuntu-b:~$ rm -rf /var/lib/postgresql/15/*
[postgres@ubuntu-b:~$ ls /var/lib/po
 polkit-1/   postgresql/
[postgres@ubuntu-b:~$ ls /var/lib/postgresql/15/
[postgres@ubuntu-b:~$ mkdir /var/lib/postgresql/15/main
[postgres@ubuntu-b:~$ chmod go-rwx /var/lib/postgresql/15/main/
```

pg_basebackup -h 172.23.0.3 -U replica_user -X stream -C -S replica_1 -v -R -W -D /var/lib/postgresql/15/main



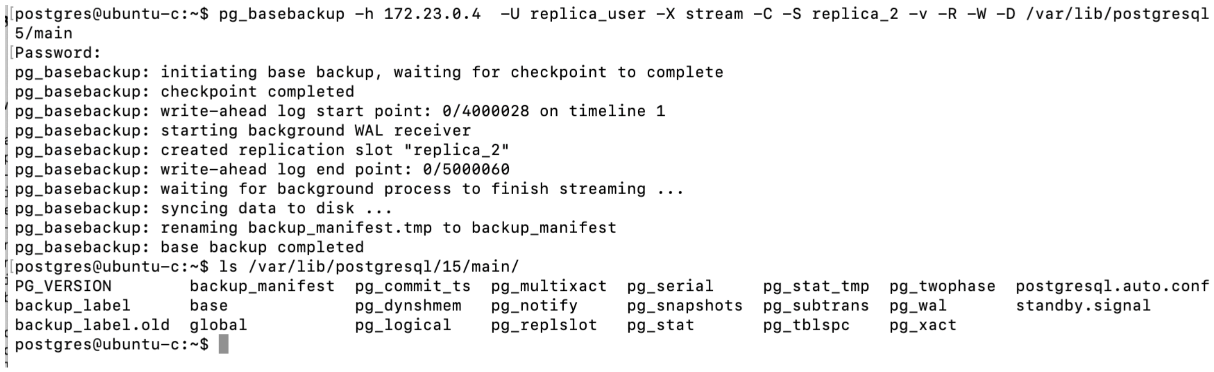
```
[postgres@ubuntu-b:~$ pg_basebackup -h 172.23.0.3  -U replica_user -X stream -C -
 S replica_1  -v -R -W -D /var/lib/postgresql/15/main
[Password:
 pg_basebackup: initiating base backup, waiting for checkpoint to complete
 pg_basebackup: checkpoint completed
 pg_basebackup: write-ahead log start point: 0/4000028 on timeline 1
 pg_basebackup: starting background WAL receiver
 pg_basebackup: created replication slot "replica_1"
 pg_basebackup: write-ahead log end point: 0/4000100
 pg_basebackup: waiting for background process to finish streaming ...
 pg_basebackup: syncing data to disk ...
 pg_basebackup: renaming backup_manifest.tmp to backup_manifest
 pg_basebackup: base backup completed
[postgres@ubuntu-b:~$ ls /var/lib/postgresql/15/main/
 PG_VERSION       pg_commit_ts  pg_replslot   pg_subtrans  postgresql.auto.conf
 backup_label     pg_dynshmem   pg_serial     pg_tblspc    standby.signal
 backup_manifest  pg_logical    pg_snapshots  pg_twophase
 base             pg_multixact  pg_stat       pg_wal
 global           pg_notify     pg_stat_tmp   pg_xact
```

UBUNTU-C



```
[postgres@ubuntu-c:~$ rm -rf /var/lib/postgresql/15/*
[postgres@ubuntu-c:~$ mkdir /var/lib/postgresql/15/main
[postgres@ubuntu-c:~$ chmod go-rwx /var/lib/postgresql/15/main/
```

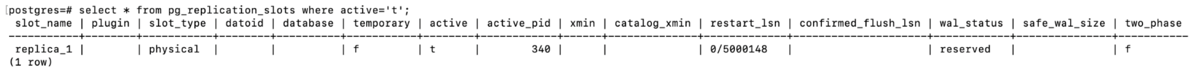
pg_basebackup -h 172.23.0.4 -U replica_user -X stream -C -S replica_2 -v -R -W -D /var/lib/postgresql/15/main



```
[postgres@ubuntu-c:~$ pg_basebackup -h 172.23.0.4  -U replica_user -X stream -C -S replica_2 -v -R -W -D /var/lib/postgresql
5/main
[Password:
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/4000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: created replication slot "replica_2"
pg_basebackup: write-ahead log end point: 0/5000060
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: renaming backup_manifest.tmp to backup_manifest
pg_basebackup: base backup completed
[postgres@ubuntu-c:~$ ls /var/lib/postgresql/15/main/
PG_VERSION        backup_manifest  pg_commit_ts  pg_multixact  pg_serial     pg_stat_tmp  pg_twophase  postgresql.auto.conf
backup_label      base             pg_dynshmem   pg_notify     pg_snapshots  pg_subtrans  pg_wal       standby.signal
backup_label.old  global           pg_logical    pg_replslot   pg_stat       pg_tblspc    pg_xact
postgres@ubuntu-c:~$ 
```

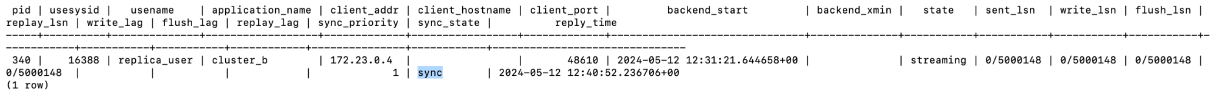
Результат

На узле A видим, что репликация идет на узел B:



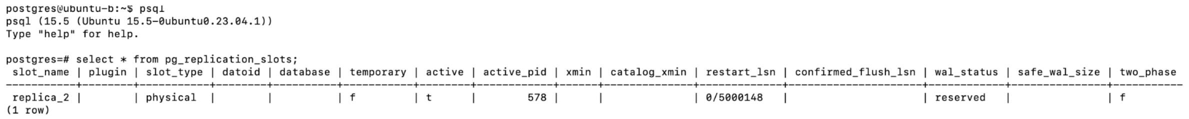
```
[postgres=# select * from pg_replication_slots where active='t';
 slot_name | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn | wal_status | safe_wal_size | two_phase
-----------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------+------------+---------------+-----------
 replica_1 |        | physical  |        |          | f         | t      |        340 |      |              | 0/5000148   |                     | reserved   |               | f
(1 row)
```

Кластер B подключен синхронно:



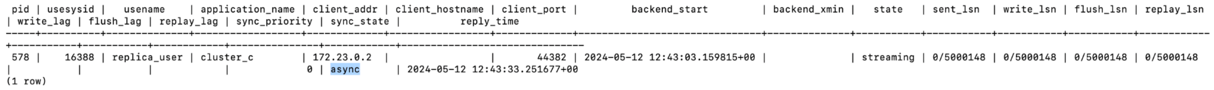
```
 pid | usesysid |   usename    | application_name | client_addr | client_hostname | client_port |         backend_start         | backend_xmin |   state   | sent_lsn  | write_lsn | flush_lsn |
replay_lsn | write_lag | flush_lag | replay_lag | sync_priority | sync_state |          reply_time
-----+----------+--------------+------------------+-------------+-----------------+-------------+-------------------------------+--------------+-----------+-----------+-----------+-----------+-
-----------+-----------+-----------+------------+---------------+------------+-------------------------------
 340 |    16388 | replica_user | cluster_b        | 172.23.0.4  |                 |       48610 | 2024-05-12 12:31:21.644658+00 |              | streaming | 0/5000148 | 0/5000148 | 0/5000148 |
0/5000148  |           |           |            |             1 | sync       | 2024-05-12 12:40:52.236706+00
(1 row)
```

На узле B репликация каскадно продолжается на C:



```
postgres@ubuntu-b:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# select * from pg_replication_slots;
 slot_name | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn | wal_status | safe_wal_size | two_phase
-----------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------+------------+---------------+-----------
 replica_2 |        | physical  |        |          | f         | t      |        578 |      |              | 0/5000148   |                     | reserved   |               | f
(1 row)
```

Кластер C подключен асинхронно:



```
 pid | usesysid |   usename    | application_name | client_addr | client_hostname | client_port |         backend_start         | backend_xmin |   state   | sent_lsn  | write_lsn | flush_lsn | replay_lsn
| write_lag | flush_lag | replay_lag | sync_priority | sync_state |          reply_time
-----+----------+--------------+------------------+-------------+-----------------+-------------+-------------------------------+--------------+-----------+-----------+-----------+-----------+------------
+-----------+-----------+------------+---------------+------------+-------------------------------
 578 |    16388 | replica_user | cluster_c        | 172.23.0.2  |                 |       44382 | 2024-05-12 12:43:03.159815+00 |              | streaming | 0/5000148 | 0/5000148 | 0/5000148 | 0/5000148
|           |           |            |             0 | async      | 2024-05-12 12:43:33.251677+00
(1 row)
```

pgpool

Добавим информацию о подключении в файл pool_hba.conf:

```
host    all         all         172.23.0.0/24         trust
```

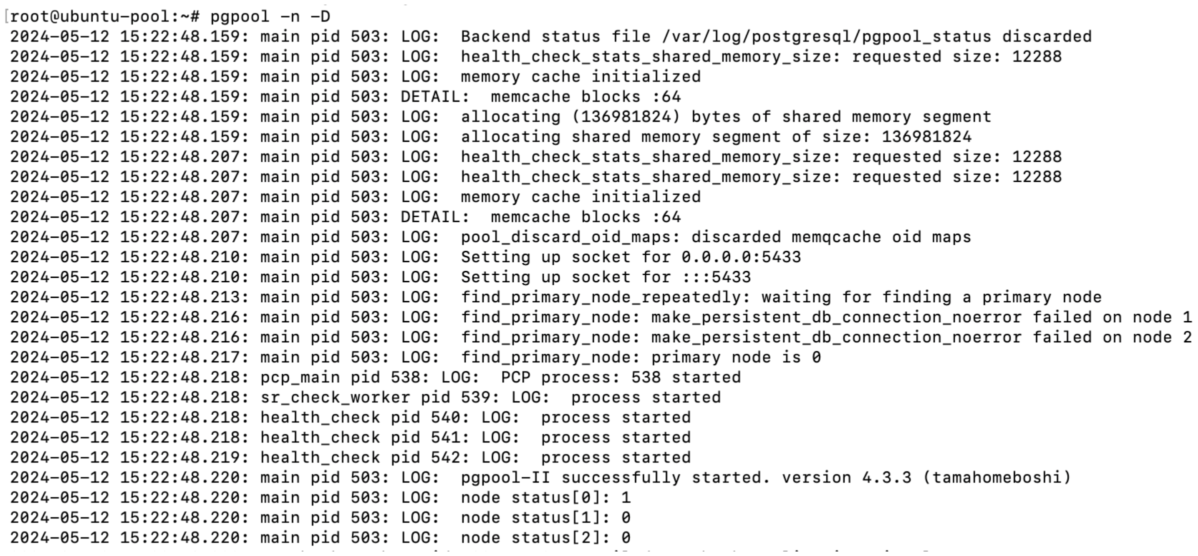
Теперь внесем изменения в pgpool.conf:

```
backend_hostname0 = '172.23.0.3'
                                    # Host name or IP address to connect to for backend 0
backend_port0 = 5432
                                    # Port number for backend 0
backend_weight0 = 1
```

```
                      # Weight for backend 0 (only in load balancing mode)
backend_data_directory0 = '/var/lib/postgresql/15/main'
                      # Data directory for backend 0
backend_flag0 = 'ALLOW_TO_FAILOVER'
                      # Controls various backend behavior
                      # ALLOW_TO_FAILOVER, DISALLOW_TO_FAILOVER
                      # or ALWAYS_PRIMARY
backend_application_name0 = 'node_a'
                      # walsender's application_name, used for "show pool_nodes" command
backend_hostname1 = '172.23.0.4'
backend_port1 = 5432
backend_weight1 = 1
backend_data_directory1 = '/var/lib/postgresql/15/main'
backend_flag1 = 'ALLOW_TO_FAILOVER'
backend_application_name1 = 'node_b'
backend_hostname2 = '172.23.0.2'
backend_port2 = 5432
backend_weight2 = 1
backend_data_directory2 = '/var/lib/postgresql/15/main'
backend_flag2 = 'ALLOW_TO_FAILOVER'
backend_application_name2 = 'node_c'


failover_when_quorum_exists = off
enable_pool_hba = on
sr_check_user = 'postgres'
sr_check_database = 'postgres'
```

Запускаем pgpool:

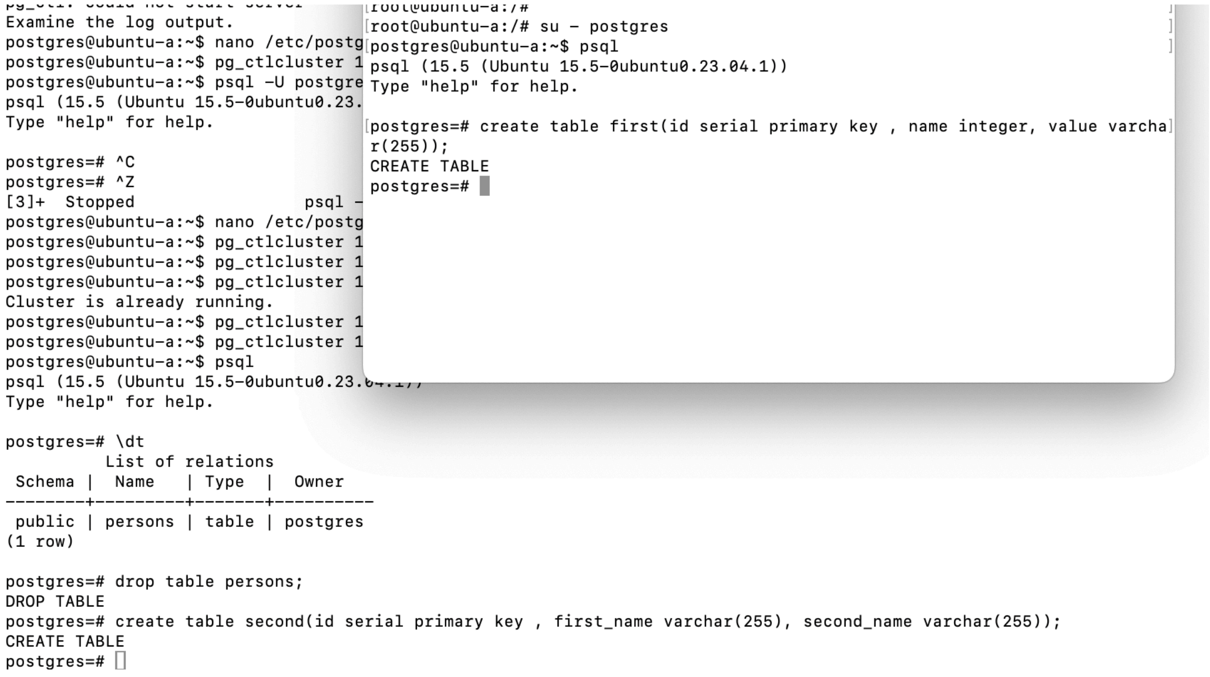
pgpool -n -D



```
root@ubuntu-pool:~# pgpool -n -D
 2024-05-12 15:22:48.159: main pid 503: LOG:  Backend status file /var/log/postgresql/pgpool_status discarded
 2024-05-12 15:22:48.159: main pid 503: LOG:  health_check_stats_shared_memory_size: requested size: 12288
 2024-05-12 15:22:48.159: main pid 503: LOG:  memory cache initialized
 2024-05-12 15:22:48.159: main pid 503: DETAIL:  memcache blocks :64
 2024-05-12 15:22:48.159: main pid 503: LOG:  allocating (136981824) bytes of shared memory segment
 2024-05-12 15:22:48.159: main pid 503: LOG:  allocating shared memory segment of size: 136981824
 2024-05-12 15:22:48.207: main pid 503: LOG:  health_check_stats_shared_memory_size: requested size: 12288
 2024-05-12 15:22:48.207: main pid 503: LOG:  health_check_stats_shared_memory_size: requested size: 12288
 2024-05-12 15:22:48.207: main pid 503: LOG:  memory cache initialized
 2024-05-12 15:22:48.207: main pid 503: DETAIL:  memcache blocks :64
 2024-05-12 15:22:48.207: main pid 503: LOG:  pool_discard_oid_maps: discarded memqcache oid maps
 2024-05-12 15:22:48.210: main pid 503: LOG:  Setting up socket for 0.0.0.0:5433
 2024-05-12 15:22:48.210: main pid 503: LOG:  Setting up socket for :::5433
 2024-05-12 15:22:48.213: main pid 503: LOG:  find_primary_node_repeatedly: waiting for finding a primary node
 2024-05-12 15:22:48.216: main pid 503: LOG:  find_primary_node: make_persistent_db_connection_noerror failed on node 1
 2024-05-12 15:22:48.216: main pid 503: LOG:  find_primary_node: make_persistent_db_connection_noerror failed on node 2
 2024-05-12 15:22:48.217: main pid 503: LOG:  find_primary_node: primary node is 0
 2024-05-12 15:22:48.218: pcp_main pid 538: LOG:  PCP process: 538 started
 2024-05-12 15:22:48.218: sr_check_worker pid 539: LOG:  process started
 2024-05-12 15:22:48.218: health_check pid 540: LOG:  process started
 2024-05-12 15:22:48.218: health_check pid 541: LOG:  process started
 2024-05-12 15:22:48.219: health_check pid 542: LOG:  process started
 2024-05-12 15:22:48.220: main pid 503: LOG:  pgpool-II successfully started. version 4.3.3 (tamahomeboshi)
 2024-05-12 15:22:48.220: main pid 503: LOG:  node status[0]: 1
 2024-05-12 15:22:48.220: main pid 503: LOG:  node status[1]: 0
 2024-05-12 15:22:48.220: main pid 503: LOG:  node status[2]: 0
```

Этап 2.1. Подготовка

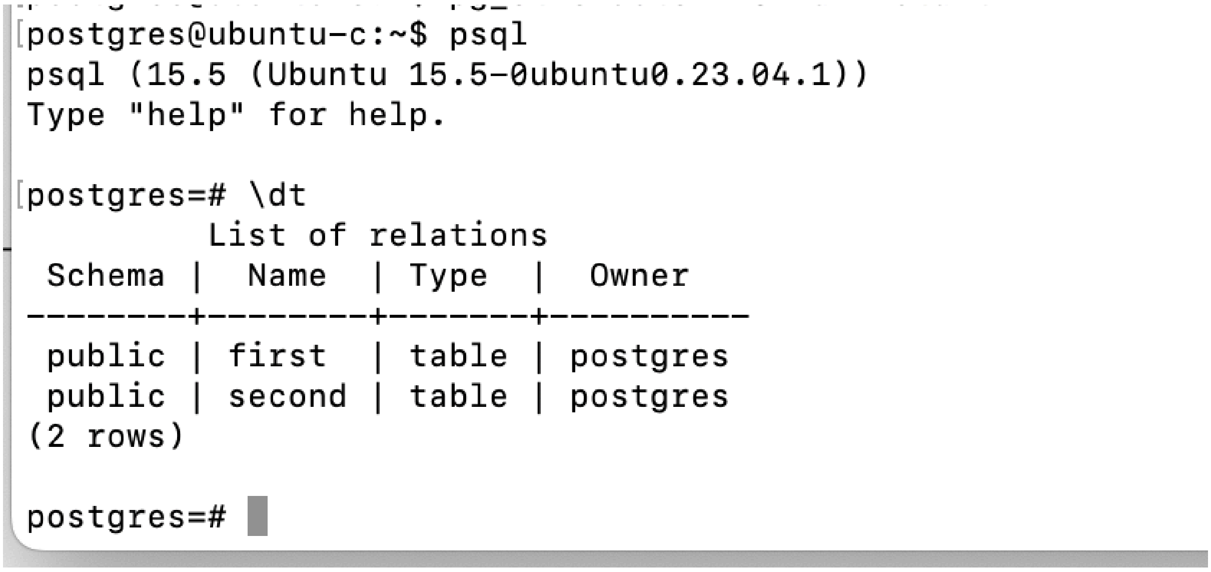
Создаем таблицы с двух сессий:



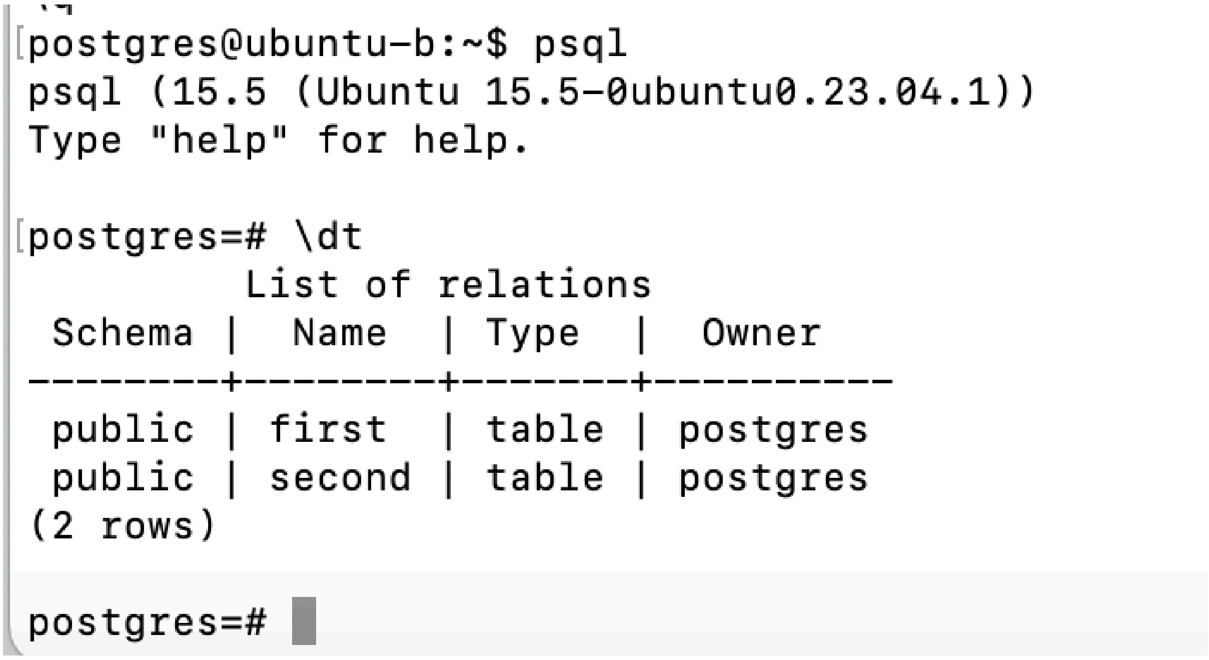
```
pg_ctl: could not start server
Examine the log output.
postgres@ubuntu-a:~$ nano /etc/postg
postgres@ubuntu-a:~$ pg_ctlcluster 1
postgres@ubuntu-a:~$ psql -U postgre
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.
Type "help" for help.

postgres=# ^C
postgres=# ^Z
[3]+  Stopped                 psql -
postgres@ubuntu-a:~$ nano /etc/postg
postgres@ubuntu-a:~$ pg_ctlcluster 1
postgres@ubuntu-a:~$ pg_ctlcluster 1
postgres@ubuntu-a:~$ pg_ctlcluster 1
Cluster is already running.
postgres@ubuntu-a:~$ pg_ctlcluster 1
postgres@ubuntu-a:~$ pg_ctlcluster 1
postgres@ubuntu-a:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)

postgres=# drop table persons;
DROP TABLE
postgres=# create table second(id serial primary key , first_name varchar(255), second_name varchar(255));
CREATE TABLE
postgres=# 
```

```
root@ubuntu-a:/#
root@ubuntu-a:/# su - postgres
postgres@ubuntu-a:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# create table first(id serial primary key , name integer, value varcha
r(255));
CREATE TABLE
postgres=# 
```

Проверим, что на узлах тоже появились таблицы:

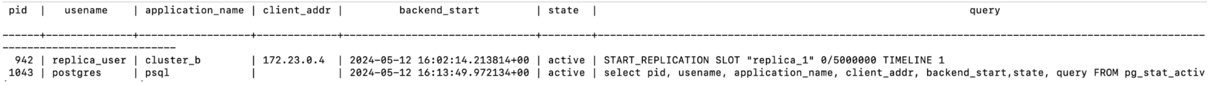


```
postgres@ubuntu-c:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | first  | table | postgres
 public | second | table | postgres
(2 rows)

postgres=#
```



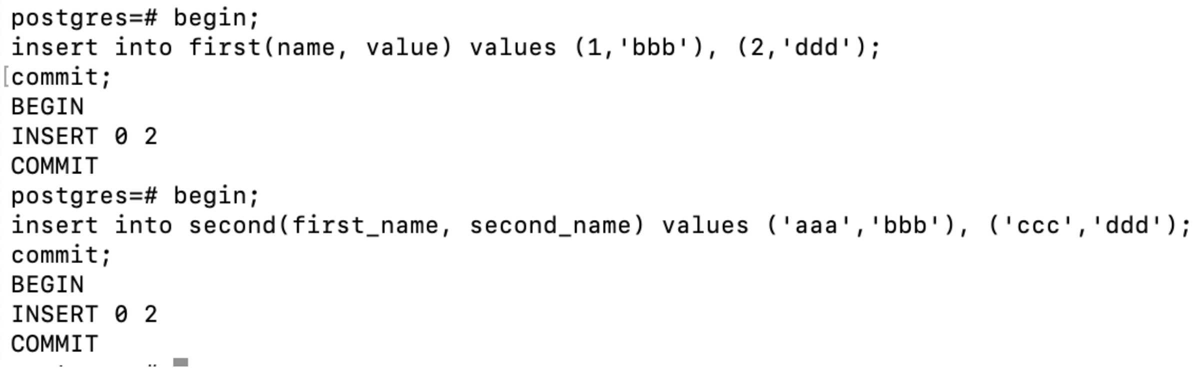
```
postgres@ubuntu-b:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | first  | table | postgres
 public | second | table | postgres
(2 rows)

postgres=#
```

Активные сессии на узле A:



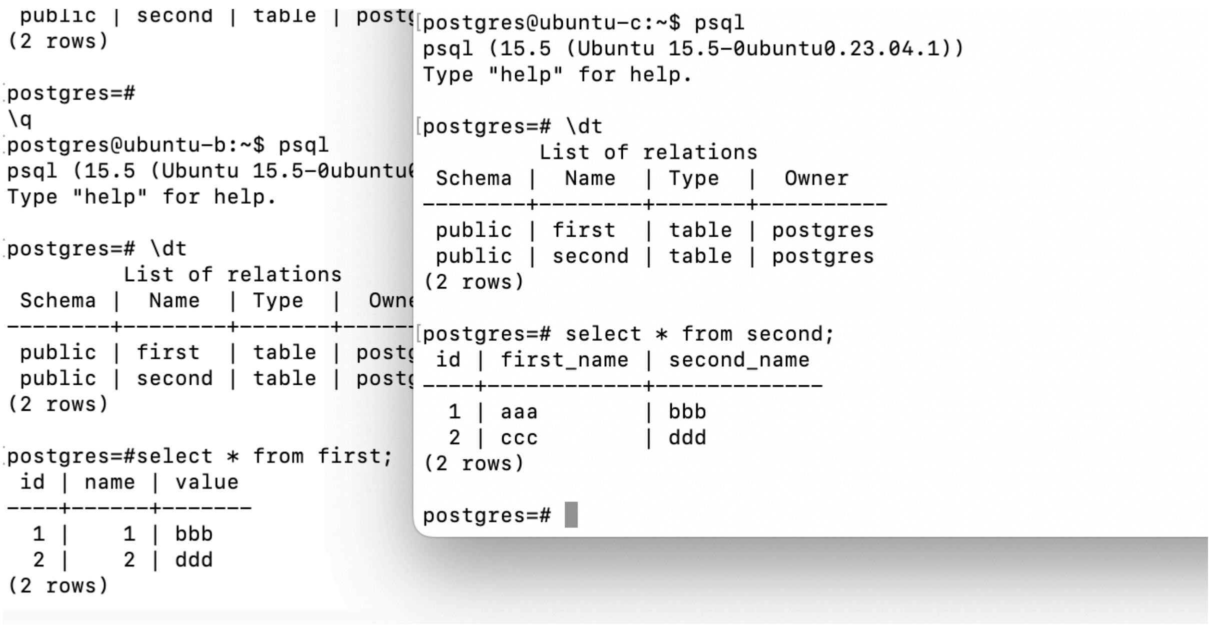
```
 pid  |   usename    | application_name | client_addr |         backend_start         | state  |                                          query
------+--------------+------------------+-------------+-------------------------------+--------+----------------------------------------------------------------------------------------------------------
----------------------------
  942 | replica_user | cluster_b        | 172.23.0.4  | 2024-05-12 16:02:14.213814+00 | active | START_REPLICATION SLOT "replica_1" 0/5000000 TIMELINE 1
 1043 | postgres     | psql             |             | 2024-05-12 16:13:49.972134+00 | active | select pid, usename, application_name, client_addr, backend_start,state, query FROM pg_stat_activ
```

Вставка данных в таблицы:



```
postgres=# begin;
insert into first(name, value) values (1,'bbb'), (2,'ddd');
commit;
BEGIN
INSERT 0 2
COMMIT
postgres=# begin;
insert into second(first_name, second_name) values ('aaa','bbb'), ('ccc','ddd');
commit;
BEGIN
INSERT 0 2
COMMIT
```

Проверка вставки:



```
 public | second | table | postg
(2 rows)

postgres=#
\q
postgres@ubuntu-b:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu
Type "help" for help.

postgres=# \dt
         List of relations
 Schema |  Name  | Type  |  Owne
--------+--------+-------+------
 public | first  | table | postg
 public | second | table | postg
(2 rows)

postgres=#select * from first;
 id | name | value
----+------+-------
  1 |    1 | bbb
  2 |    2 | ddd
(2 rows)
```

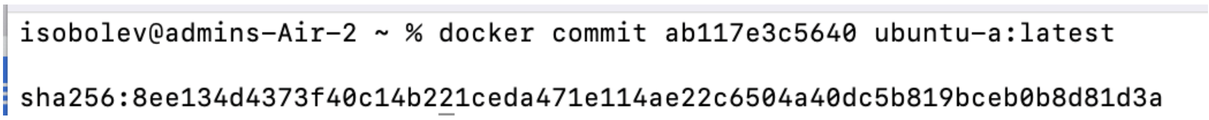
```
postgres@ubuntu-c:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | first  | table | postgres
 public | second | table | postgres
(2 rows)

postgres=# select * from second;
 id | first_name | second_name
----+------------+-------------
  1 | aaa        | bbb
  2 | ccc        | ddd
(2 rows)

postgres=#
```

Как мы можем увидеть, данные реплицируются по узлам.

Этап 2.2. Сбой

Создаем снимок контейнера:

docker commit ab117e3c5640 ubuntu-a:latest



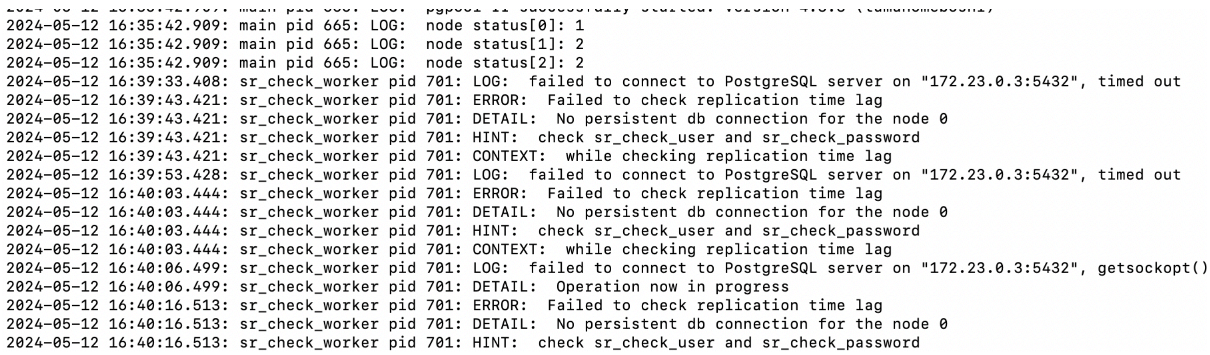
```
isobolev@admins-Air-2 ~ % docker commit ab117e3c5640 ubuntu-a:latest

sha256:8ee134d4373f40c14b221ceda471e114ae22c6504a40dc5b819bceb0b8d81d3a
```

Эмулируем сбой основного узла:

docker stop -s=9 ab117e3c5640 # sigkill

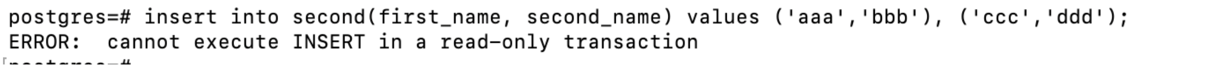
Этап 2.3. Отработка

Логи об ошибках:



```
2024-05-12 16:35:42.909: main pid 665: LOG:  pgpool-II successfully started. version 4.3.5 (tamahomeboshi)
2024-05-12 16:35:42.909: main pid 665: LOG:  node status[0]: 1
2024-05-12 16:35:42.909: main pid 665: LOG:  node status[1]: 2
2024-05-12 16:35:42.909: main pid 665: LOG:  node status[2]: 2
2024-05-12 16:39:33.408: sr_check_worker pid 701: LOG:  failed to connect to PostgreSQL server on "172.23.0.3:5432", timed out
2024-05-12 16:39:43.421: sr_check_worker pid 701: ERROR:  Failed to check replication time lag
2024-05-12 16:39:43.421: sr_check_worker pid 701: DETAIL:  No persistent db connection for the node 0
2024-05-12 16:39:43.421: sr_check_worker pid 701: HINT:  check sr_check_user and sr_check_password
2024-05-12 16:39:43.421: sr_check_worker pid 701: CONTEXT:  while checking replication time lag
2024-05-12 16:39:53.428: sr_check_worker pid 701: LOG:  failed to connect to PostgreSQL server on "172.23.0.3:5432", timed out
2024-05-12 16:40:03.444: sr_check_worker pid 701: ERROR:  Failed to check replication time lag
2024-05-12 16:40:03.444: sr_check_worker pid 701: DETAIL:  No persistent db connection for the node 0
2024-05-12 16:40:03.444: sr_check_worker pid 701: HINT:  check sr_check_user and sr_check_password
2024-05-12 16:40:03.444: sr_check_worker pid 701: CONTEXT:  while checking replication time lag
2024-05-12 16:40:06.499: sr_check_worker pid 701: LOG:  failed to connect to PostgreSQL server on "172.23.0.3:5432", getsockopt()
2024-05-12 16:40:06.499: sr_check_worker pid 701: DETAIL:  Operation now in progress
2024-05-12 16:40:16.513: sr_check_worker pid 701: ERROR:  Failed to check replication time lag
2024-05-12 16:40:16.513: sr_check_worker pid 701: DETAIL:  No persistent db connection for the node 0
2024-05-12 16:40:16.513: sr_check_worker pid 701: HINT:  check sr_check_user and sr_check_password
```

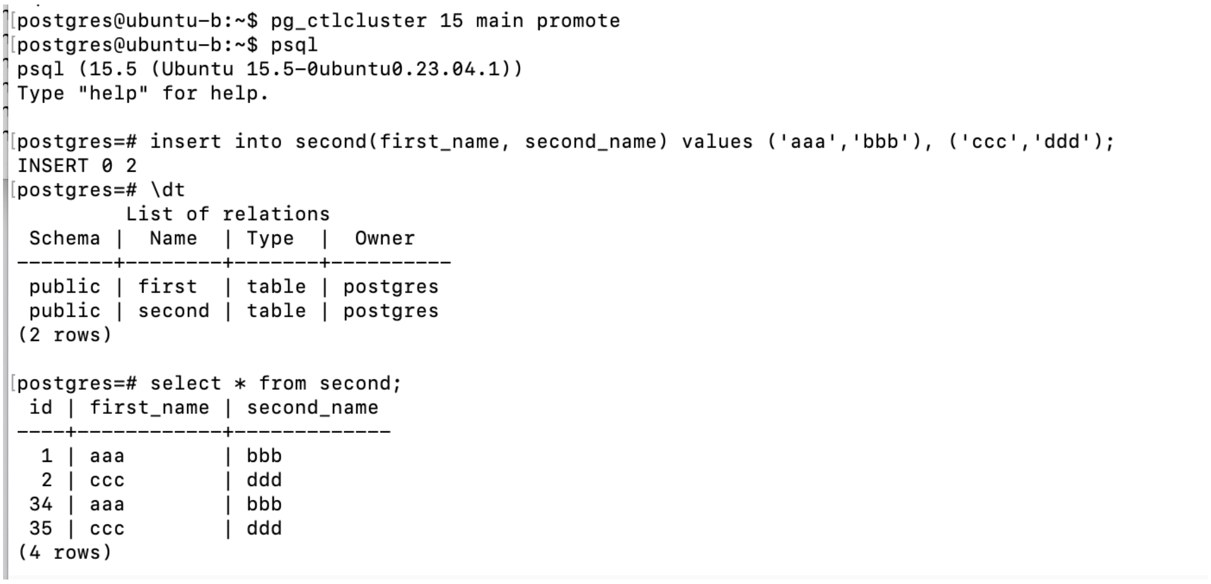
На резервном узле писать не можем:



```
postgres=# insert into second(first_name, second_name) values ('aaa','bbb'), ('ccc','ddd');
ERROR:  cannot execute INSERT in a read-only transaction
postgres=#
```

Осуществляем failover. На узле пишем:

pg_ctlcluster 15 main promote



```
postgres@ubuntu-b:~$ pg_ctlcluster 15 main promote
postgres@ubuntu-b:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# insert into second(first_name, second_name) values ('aaa','bbb'), ('ccc','ddd');
INSERT 0 2
postgres=# \dt
         List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | first  | table | postgres
 public | second | table | postgres
(2 rows)

postgres=# select * from second;
 id | first_name | second_name
----+------------+-------------
  1 | aaa        | bbb
  2 | ccc        | ddd
 34 | aaa        | bbb
 35 | ccc        | ddd
(4 rows)
```

Этап 3. Восстановление

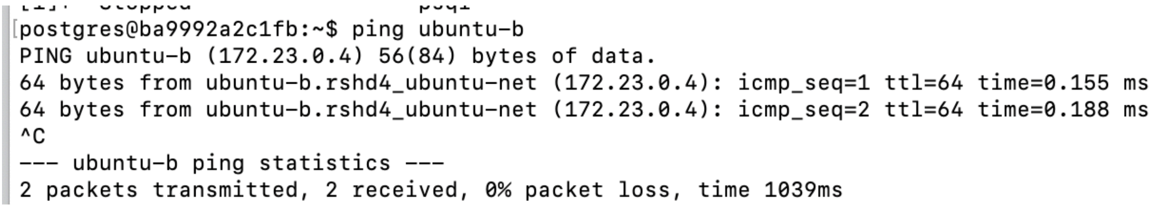
С помощью созданного коммита запускаем убитую основную бд:

docker run -it -d --network=rshd4_ubuntu-net ubuntu-a:latest

Подключаемся к контейнеру:

docker exec -it ba9992a bash

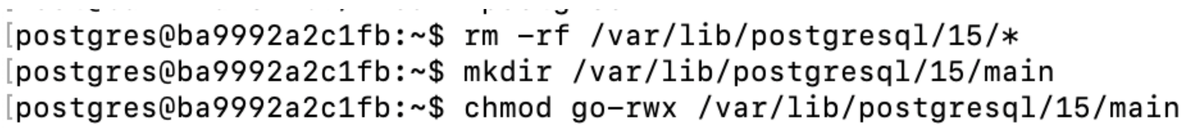
Проверяем включение в сеть:



```
[1]+  Stopped                 psql
[postgres@ba9992a2c1fb:~$ ping ubuntu-b
PING ubuntu-b (172.23.0.4) 56(84) bytes of data.
64 bytes from ubuntu-b.rshd4_ubuntu-net (172.23.0.4): icmp_seq=1 ttl=64 time=0.155 ms
64 bytes from ubuntu-b.rshd4_ubuntu-net (172.23.0.4): icmp_seq=2 ttl=64 time=0.188 ms
^C
--- ubuntu-b ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1039ms
```

Восстанавливаем ее состояние со бд, в которой уже были произведены изменения:

Удаляем PGDATA:

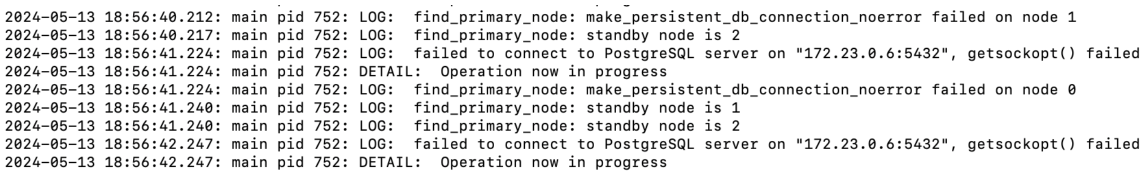


```
[postgres@ba9992a2c1fb:~$ rm -rf /var/lib/postgresql/15/*
[postgres@ba9992a2c1fb:~$ mkdir /var/lib/postgresql/15/main
[postgres@ba9992a2c1fb:~$ chmod go-rwx /var/lib/postgresql/15/main
```

Забираем данные с узла B:

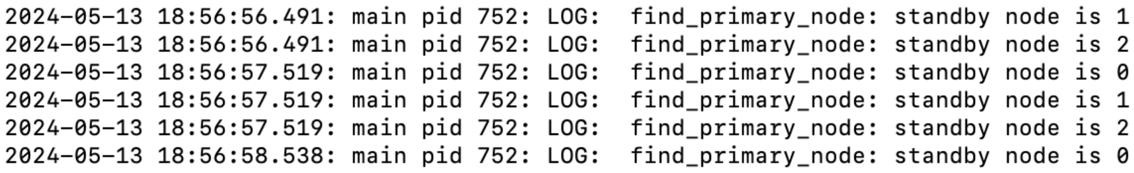
pg_basebackup -h 172.23.0.4 -U replica_user -X stream -C -S back -v -R -W -D /var/lib/postgresql/15/main

На узле B создаем файл standby.signal, чтобы запустить его в режиме слейва, останавливаем и смотрим в pgpool:



```
2024-05-13 18:56:40.212: main pid 752: LOG:  find_primary_node: make_persistent_db_connection_noerror failed on node 1
2024-05-13 18:56:40.217: main pid 752: LOG:  find_primary_node: standby node is 2
2024-05-13 18:56:41.224: main pid 752: LOG:  failed to connect to PostgreSQL server on "172.23.0.6:5432", getsockopt() failed
2024-05-13 18:56:41.224: main pid 752: DETAIL:  Operation now in progress
2024-05-13 18:56:41.224: main pid 752: LOG:  find_primary_node: make_persistent_db_connection_noerror failed on node 0
2024-05-13 18:56:41.240: main pid 752: LOG:  find_primary_node: standby node is 1
2024-05-13 18:56:41.240: main pid 752: LOG:  find_primary_node: standby node is 2
2024-05-13 18:56:42.247: main pid 752: LOG:  failed to connect to PostgreSQL server on "172.23.0.6:5432", getsockopt() failed
2024-05-13 18:56:42.247: main pid 752: DETAIL:  Operation now in progress
```

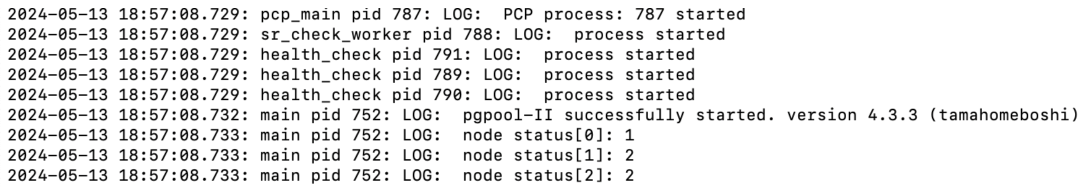
Сначала видим, что standby node только один - C, после запуска сервера B их стало две и начала падать ошибка подключения к серверу A. Включаем A:



```
2024-05-13 18:56:56.491: main pid 752: LOG:  find_primary_node: standby node is 1
2024-05-13 18:56:56.491: main pid 752: LOG:  find_primary_node: standby node is 2
2024-05-13 18:56:57.519: main pid 752: LOG:  find_primary_node: standby node is 0
2024-05-13 18:56:57.519: main pid 752: LOG:  find_primary_node: standby node is 1
2024-05-13 18:56:57.519: main pid 752: LOG:  find_primary_node: standby node is 2
2024-05-13 18:56:58.538: main pid 752: LOG:  find_primary_node: standby node is 0
```

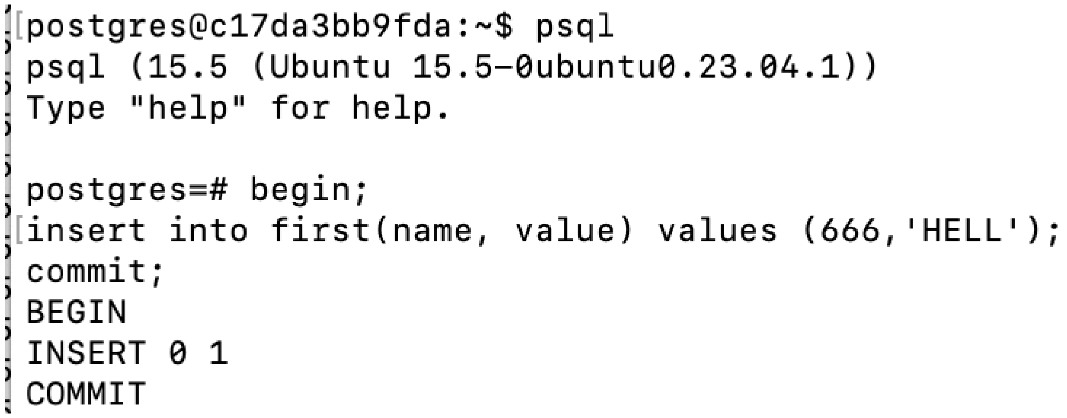
Теперь стало 3 standby ноды, осталось сделать:

pg_ctlcluster 15 main promote

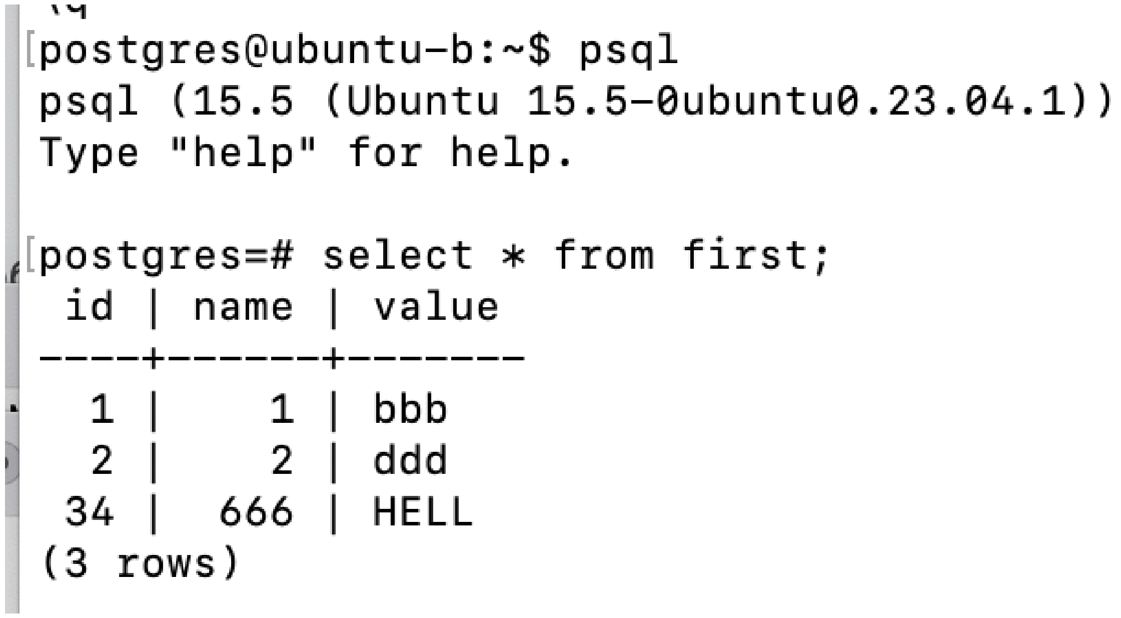


```
2024-05-13 18:57:08.729: pcp_main pid 787: LOG:  PCP process: 787 started
2024-05-13 18:57:08.729: sr_check_worker pid 788: LOG:  process started
2024-05-13 18:57:08.729: health_check pid 791: LOG:  process started
2024-05-13 18:57:08.729: health_check pid 789: LOG:  process started
2024-05-13 18:57:08.729: health_check pid 790: LOG:  process started
2024-05-13 18:57:08.732: main pid 752: LOG:  pgpool-II successfully started. version 4.3.3 (tamahomeboshi)
2024-05-13 18:57:08.733: main pid 752: LOG:  node status[0]: 1
2024-05-13 18:57:08.733: main pid 752: LOG:  node status[1]: 2
2024-05-13 18:57:08.733: main pid 752: LOG:  node status[2]: 2
```

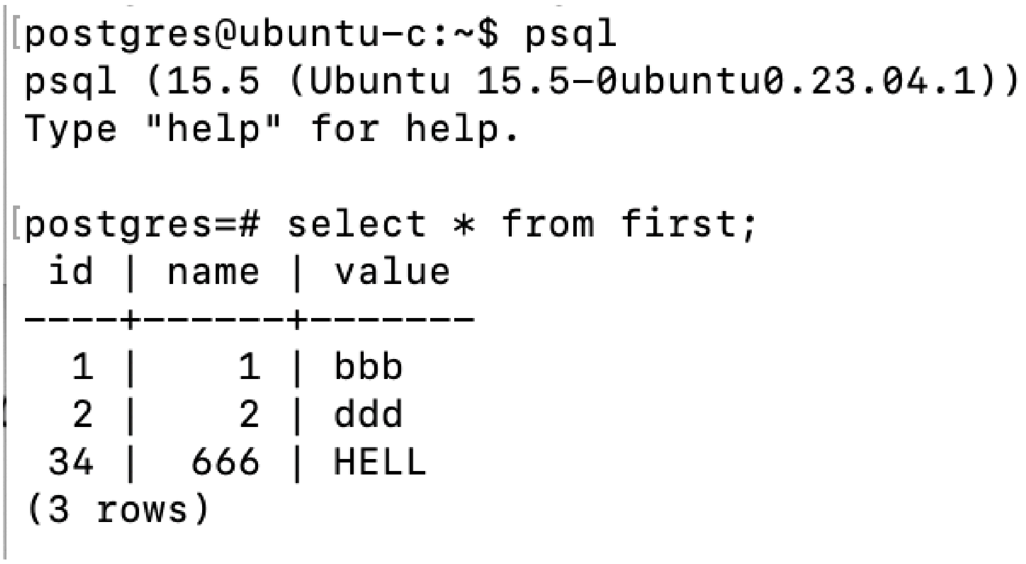
Видим, что мастером стал сервер A. Проверим репликацию:



```
postgres@c17da3bb9fda:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# begin;
insert into first(name, value) values (666,'HELL');
commit;
BEGIN
INSERT 0 1
COMMIT
```



```
postgres@ubuntu-b:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# select * from first;
 id | name | value
----+------+-------
  1 |    1 | bbb
  2 |    2 | ddd
 34 |  666 | HELL
(3 rows)
```



```
postgres@ubuntu-c:~$ psql
psql (15.5 (Ubuntu 15.5-0ubuntu0.23.04.1))
Type "help" for help.

postgres=# select * from first;
 id | name | value
----+------+-------
  1 |    1 | bbb
  2 |    2 | ddd
 34 |  666 | HELL
(3 rows)
```

Все вернулось в исходное состояние!

Выводы

В ходе выполнения данной работы я научился строить отказоустойчивые решения на базе СУБД Postgres, получил практические навыки восстановления работы системы после отказа.

Также научился работать с pgpool-II с целью автоматического назначения новой мастер ноды, если текущая не способна работать корректно.

Кроме того, разобрался с различными видами сетей в docker, а также узнал про команду docker commit для создания image с текущим состоянием контейнера.